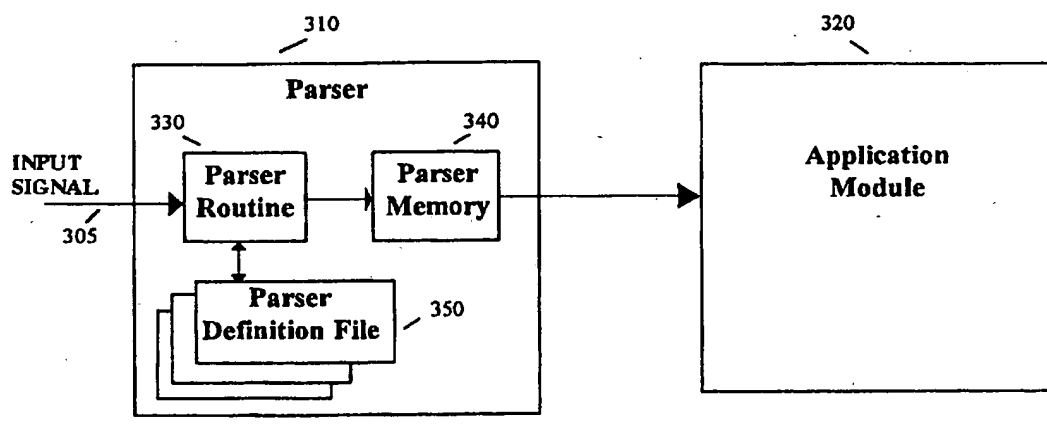


PCTWORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F 9/44	A1	(11) International Publication Number: WO 97/08616 (43) International Publication Date: 6 March 1997 (06.03.97)
(21) International Application Number: PCT/US96/09821 (22) International Filing Date: 11 June 1996 (11.06.96) (30) Priority Data: 08/520,526 29 August 1995 (29.08.95) US (71) Applicant: BELL COMMUNICATIONS RESEARCH, INC. [US/US]; 445 South Street, Morristown, NJ 07960-6438 (US). (72) Inventors: MARATHE, Rohini, V.; 11 Sherwood Court, Holmdel, NJ 07733 (US). SHASTRY, Subramanya; 66 Green Meadow Boulevard, Middletown, NJ 07748 (US). (74) Agents: GIORDANO, Joseph et al.; International Coordinator, Room 1G112R, 445 South Street, Morristown, NJ 07960-6438 (US).		(81) Designated States: CA, CN, JP, KR, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE). Published <i>With international search report.</i>
(54) Title: SYSTEM AND METHOD FOR PARSING AND BUILDING DATA SIGNALS  <pre>graph LR subgraph 310 [310] direction TB subgraph Parser direction LR PR[Parser Routine 330] PM[Parser Memory 340] PR --> PM end PR <--> PDF[Parser Definition File 350] end IS[INPUT SIGNAL 305] --> PR PM --> 320 subgraph 320 [320] direction TB AM[Application Module] end</pre>		
(57) Abstract An automatic parser (310) receives a data stream (305) and decomposes it into individual pieces of data based on decomposing information stored in a definition file (350) and transmits the individual pieces of data into a memory (340) based on the decomposing information. The parser can also detect errors in an encoded received data stream based on error information and handle any detected errors. The parser waits to detect all of the errors in the received data stream before it begins to handle any of the errors. By storing a plurality of separate sets of parsing information, each corresponding to a different format, the parser can decompose messages from a plurality of formats by initially determining the format of the received data stream and using the appropriate format. A builder can also be used to perform this process in reverse.		

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgyzstan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic	KR	Republic of Korea	SE	Sweden
CG	Congo	KZ	Kazakhstan	SG	Singapore
CH	Switzerland	LI	Liechtenstein	SI	Slovenia
CI	Côte d'Ivoire	LK	Sri Lanka	SK	Slovakia
CM	Cameroon	LR	Liberia	SN	Senegal
CN	China	LT	Lithuania	SZ	Swaziland
CS	Czechoslovakia	LU	Luxembourg	TD	Chad
CZ	Czech Republic	LV	Latvia	TG	Togo
DE	Germany	MC	Monaco	TJ	Tajikistan
DK	Denmark	MD	Republic of Moldova	TT	Trinidad and Tobago
EE	Estonia	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	UG	Uganda
FI	Finland	MN	Mongolia	US	United States of America
FR	France	MR	Mauritania	UZ	Uzbekistan
GA	Gabon			VN	Viet Nam

SYSTEM AND METHOD FOR PARSING AND BUILDING DATA SIGNALS

BACKGROUND OF THE INVENTION

The present invention relates generally to techniques for parsing data signals, and more specifically to parsing data signals transmitted between modules in a telecommunications network.

As with any network, a telecommunications network passes information between modules within the network via data signals. The signals, which are simply streams of data bits, require some method of organization so that the network modules can understand and communicate the data. Signals passing between modules of a telecommunications system are formatted according to data formats. One data format, the Binary Encoding Rules ("BER") format, organizes data into bytes of eight bits and specifies the order of the various bytes. BER is a machine-readable form of the human-readable format of Abstract Syntax Notation ("ASN.1").

Typically, source modules use builders to format groups of data bytes so that they may be transmitted as decipherable signals. Destination modules use parsers, coordinated in format with the builders, to extract the data bytes from the transmitted signals.

The messages are specified using ASN.1 in interface documents between two modules. Computer programs are then developed to translate the messages that adhere to the ASN.1 specification into BER format for transmission between modules in the network. The ASN.1 specification language and the BER format description can be found in "THE OPEN BOOK: A Practical Perspective on OSI", by Marshall T. Rose (Prentice Hall 1990).

Different types of messages may have different types of ASN.1 specifications, resulting in different BER streams. Examples of , ASN.1 message specifications are AIN-0 protocol specification and AIN-0.1 protocol specification. Each specification requires that its own parsing routine and building routine be compiled.

For each message specification, AIN-0 and AIN-0.1, for example, the builder must create and compile separate parsing routines and building routines. In addition, if new specifications are introduced, the user of the network must create and compile new parsing and building routines for each specification and integrate these new routines into each specification. This makes it very complicated to add new messages or protocol specifications to the network.

Fig. 1A is a block diagram showing an exemplary background parser 110 and Fig. 1B is a block diagram showing an exemplary background builder 115. As shown in Fig. 1A, the parser 110 is connected to an application module 120 and is used to parse an input signal 105. The parser 110 includes at least one parsing routine 130 and at least one parsing memory 140. In operation, the parser 110 accepts the input data from the input-data signal byte-by-byte and, based on the order of the incoming bytes and the instructions contained in the parsing routine 130, places the bytes into portions of the parsing memory 140. The application module 120 then extracts the data from the portions of the parsing memory 140. As shown in Fig. 1A, the parser 110 contains one parse route for each supported message.

Fig. 2A is a flowchart of the operation of the parser 110 of Fig. 1A. The parser 110 begins by receiving the current byte of data from the input signal (step 200). The parsing routine 130 then detects whether there is any error in the received byte of data (step 205) and, if it detects an error, exits to an error handling routine for processing the detected error (step 210). If the parsing routine 130 detects no error in the received byte of data, it places the received byte of data into the portion of the parsing memory 140 corresponding to that data byte (step 215). The parsing routine 130 is programmed with the appropriate memory locations corresponding to each of the data bytes contained in the input signal.

After processing the received byte of data, the parsing routine 130 determines whether the last byte of the input signal has been received (step 220). If the last byte has not been received, the parsing routine 130 moves on to the next byte in the input signal (step 225) and returns to step 200. If the last byte of the input signal has been received, the parsing routine 130 reports successful parsing to the application module 120 (step 230) and ends processing.

As shown in Fig. 1B, the builder 115 is connected to the application module 120 and is used to construct an output signal for transmission. The builder 115 includes a building routine 135 and a building memory 145. In operation, the application module 120 places bytes of data to be transmitted within the network into portions of the building memory 145. The builder 115 then extracts the bytes of data from the building memory 145 and, based

on the memory locations of the byte of data and instructions contained in the building routine 135, arranges the bytes into an output signal and transmits the output signal. Also, as shown in Fig. 2B, builder 115 contains one build routine for each supported message.

Fig. 2B is a flowchart of the operation of the builder 115 of Fig. 1B. As shown in Fig. 2B, the builder 115 begins by extracting from the building memory 145 the byte of data corresponding to the first byte of data for the output signal (step 250). The building routine 135 then determines whether there is any error in the extracted byte of data (step 255) and, if it detects an error, exits to an error handling routine for processing the detected error (step 260). If it detects no error in the extracted byte of data, the building routine 135 stores the extracted byte of data as the first byte of the transmitted signal (step 265). The building routine is programmed with the appropriate memory location for each of the data bytes required to construct the data signal.

After processing the extracted byte of data, the building routine 135 determines whether the last byte required for the output signal has been extracted (step 270). If the last byte has not been extracted, the building routine 135 moves on to the next byte for the output signal (step 275) and returns to step 250. If the last byte of the incoming signal has been extracted, the building routine 135 transmits the output signal (step 280) and ends processing.

The application module 120 is blind to the data format being used. The only information available to the application module is where in the parser memory 140 each byte of the incoming signal will be placed and where in the building memory 145 each byte for the outgoing signal will be placed. The parsing routine 130 contains the information necessary to properly route the incoming data bytes into their proper locations in the parser memory 140. Similarly, the building routine 135 contains the information necessary to properly extract the outgoing data bytes from the proper locations in the building memory 145 and arrange the bytes in the proper order according to the output signal's required format.

As an example, if a particular interface between two modules is defined using 20 data formats, then 20 parse routines and 20 build routines will have to be provided to support that interface. In addition, any time the data format changes, the corresponding parse and build routines will also have to be changed.

Since each parsing routine 130 and building routine 135 is specifically designed to decipher a given data format, a separate parsing routine 130 and building routine 135 are required for each new data format that may be used by the application module 120.

The parsing and building routines can be a set of computer programs using a programming language such as C. Each parsing routine implements the logic shown in Fig. 2A and each build routine implements the logic shown in Fig. 2B. The parsing memory 140 may be a data structure, the elements of which are referenced by the parsing routine 130 and the application module 120.

Similarly, the building memory 145 may be a data structure, the elements of which are referenced by the building routine 135 and the application module 120.

Many parsing and building routines use the commercially available ASN.1 compilers (e.g., DSET compiler supported by DSET Corporation), to translate the BER input signal to a set of data structures, which in turn are used by the application module 120. These compilers use the ASN.1 specification as input and produces the parsing and building routines 140 and 145 and the data structure formats. If the ASN.1 specification is changed, then it is necessary to regenerate the parsing and building routines 140 and 145 and to change the application module 120 to reference the newly changed data structures.

In addition, because the parser 110 or builder 115 stops processing and enters an error handling routine immediately upon detecting an error, if an input signal or a builder memory 145 contains multiple errors, the parser 110 or builder 115 stops upon encountering the first error which is not the behavior expected of a network application.

DESCRIPTION OF THE INVENTION

Accordingly, the present invention is directed to a system and method for parsing or building a data signal that substantially obviates one or more of the problems due to limitations and disadvantages of the related art.

The present invention overcomes the limitations and disadvantages noted above through the use of definition files that

contain information relating to one or more data formats and information on various types of error handling. This allows the use of generic parsing and building routines that obtain any data particular to a given data format from the definition files. This also allows the parsing and building routine to identify all of the errors associated with an input or output signal before any error handling routine is invoked.

Additional features and advantages of the invention will be set forth in the description which follows, and in part will be apparent from the description, or may be learned by practice of the invention. The objectives and other advantages of the invention will be realized and attained by means of the instrumentalities and combinations particularly pointed out in the written description and appended claims hereof as well as the appended drawings.

To overcome the disadvantages of the prior art and in accordance with the purposes of the invention, as embodied and broadly described, the invention includes, an automatic parser for parsing a data stream, comprising: a memory containing a definition file; means for receiving the data stream; means for decomposing the received data stream into individual pieces of data based on parsing information in the definition file; and means for transmitting the individual pieces of data.

Also in accordance with the purposes of the invention, as embodied and broadly described, the invention includes, an automatic builder for creating an output data stream, comprising: a memory containing a definition file; means for receiving a

plurality of individual pieces of data; and means for composing the output data stream out of the individual pieces of data based on composing information in the definition file.

The invention also includes a method for automatically parsing an input data stream, including the steps, executed by a processor, of: receiving the input data stream; decomposing the received input data stream into individual pieces of data based on decomposing information stored in a definition file residing in a memory; and transmitting the individual pieces of data.

In addition, the invention includes a method for automatically creating an output data stream, including the steps, executed by a processor, of: receiving a plurality of individual pieces of data; composing the output data stream out of the individual pieces of data based on composing information stored in a definition file residing in a memory; and transmitting the composed output data stream.

Also in accordance with the purposes of the invention, as embodied and broadly described, the invention recites, an automatic parser for parsing a received data stream and creating an output data stream, comprising: a memory containing an input definition file; a memory containing an output definition file; means for decomposing the received data stream into a plurality of individual pieces of input data based on information in the input definition file; means for transmitting the individual pieces of data; means for receiving a plurality of individual pieces of output data; and means for composing the output data stream out of

the plurality of individual pieces of output data based on information in the output definition file.

It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory and are intended to provide further explanation of the invention as claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate presently preferred implementations of the invention and, together with the general description given above and the detailed description of the preferred implementations given below, serve to explain the principles of the invention.

Fig. 1A is a block diagram showing a prior art parser;

Fig. 1B is a block diagram showing a prior art builder;

Fig. 2A is a flowchart showing the steps performed by a parsing routine in parsing a data signal;

Fig. 2B is a flowchart showing the steps performed by a building routine in composing a data signal;

Fig. 3A is a block diagram showing a parser according to a preferred embodiment of the current invention;

Fig. 3B is a block diagram showing a builder according to a preferred embodiment of the current invention;

Fig. 4A is a flowchart showing the steps performed by a parsing routine in parsing a data signal according to a preferred embodiment of the current invention; and

Fig. 4B is a flowchart showing the steps performed by a building routine in composing a data signal according to a preferred embodiment of the current invention.

BEST MODE FOR CARRYING OUT THE INVENTION

Reference will now be made in detail to the construction and operation of preferred implementations of the present invention which are illustrated in the accompanying drawings. In those drawings, like elements and operations are designated with the same reference numbers where possible.

The following description of the preferred implementations of the present invention is only exemplary of the invention. The present invention is not limited to these implementations, but may be realized by other implementations.

Figs. 3A and 3B are block diagrams showing a parser 310 and a builder 315, respectively, according to a preferred embodiment of the present invention. As shown in Fig. 3A, the parser 310 is connected to an application module 320 and is used to parse an input signal. The parser 310 includes a parsing routine 330, a parsing memory 340, and at least one parsing definition file 350.

Preferably the parsing routine 330 is a computer program, such as a C program, executed by a microprocessor (not shown). The parsing routine 330 may be written in any suitable programming language and may be executed by any suitable microprocessor. The parsing memory 340 may comprise any suitable computer memory device, such as IC ROM or RAM, or disk memory.

The parsing definition file 350 is preferably a data file, so that it can be easily changed using a commercially available editor. An example of such a file is shown in Appendix A1. Preferably, this data file can be automatically generated from a corresponding ASN.1 specification, shown in Appendix A2.

In operation, the parser 310 accepts the input data byte-by-byte from the input-data signal 305 and, based on the instructions contained in the parsing routine 330 and the parsing definition file 350, places the received bytes into portions of the parsing memory 340.

Fig. 4A is a flowchart of the operation of the parser 310 of Fig. 3A. the parser 310 begins by detecting the format of the input signal (step 400) and receiving the current byte of data from the input signal (step 405). The format of the input signal is normally identified by the first few bytes of data in the input signal (usually referred to as the header in the input signal). The parsing routine 330 then determines whether there is any error in the received byte of data (step 410) and, if it detects an error, stores information regarding the error according to information in the parsing definition file 350 (step 415). If the parsing routine 330 detects no error in the received byte of data, it accesses the parsing definition file 350 to determine what address in the parsing memory 340 the received byte of data corresponds to (step 420). Then, the parsing routine 330 writes the received data byte into the corresponding address of the parsing memory 340 (step 425).

After processing the received byte of data or storing error information, the parsing routine 330 determines whether the last byte of the input signal has been received (step 430). If the last byte has not been received, the parsing routine 330 moves on to the next byte of the input signal (step 435) and returns to step 405. If the last byte of the input signal has been received, the parsing routine 330 determines whether any errors have been detected during the entire processing of the signal (step 440). If an error has been detected, the parsing routine 330 exits to an error handling routine to process the detected errors (step 443). If no errors have been detected, the parsing routine 330 reports a successful parsing to the application module 120 (step 445) and ends processing.

Different levels of error checking can be performed by the parsing routine 330. Some examples include explicit error checking, implicit error checking, protocol error checking, and existency checking. In explicit error checking, a specific error routine is associated with a parameter in the input signal. An example is shown in the definition in Appendix A1 for the parameter ID. The command RANGECHK(1,100) identifies the range of values for the parameter. In implicit error checking, only certain values are allowed, depending upon the data type. For example, if the data parameter is identified as DIGITS, only valid binary coded digits (0, 1, . . . , 9, *, #) are allowed in the input signal. In protocol error checking, the input signal is checked for proper protocol standards. For example, every parameter must be encoded according to BER rules. In existency

checking, optional parameters may or may not exist in the input signal. These parameters are identified by the keyword "OPTIONAL" in the definition file.

As shown in Fig. 3B, the builder 315 is connected to the application module 120 and is used to construct an output signal 360 for transmission from the application module 120. The builder 315 includes a building routine 335, a building memory 345, and at least one building definition file 355.

Preferably the building routine 335 is a computer program, such as a C program, executed by a microprocessor (not shown). The building routine 335 may be written in any suitable programming language, and may be executed by any suitable microprocessor. The building memory 345 may comprise any suitable computer memory device, such as IC ROM or RAM, or disk memory.

The building definition file 355 is preferably also a data file similar to the parsing definition file 350. For example, the building definition file 355 may be the same as the file shown in Appendix A1. In this case, the building routine 335 constructs the output signal 360 that adheres to the BER encoding for the ASN.1 specification of the output signal shown in Appendix A2.

In operation, the application module 120 places data to be transmitted within the network into portions of the building memory 345, from which memory portions the builder 315 then extracts the data bytes. Based on building memory 345 locations, instructions contained in the building routine 135, and information in the building definition file 355, the building

routine 335 then orders the data bytes into an output signal and transmits the output signal.

Fig. 4B is a flowchart of the operation of the building routine 335 of Fig. 3B. The building routine 335 begins by determining the desired format of the output signal (step 450). The kind of format used for creating the output signal 360 is specified by the application module 320. Then the building routine 335 accesses the building definition file 355 to determine the location in the builder memory 345 corresponding to the current byte in the output signal (step 455) and extracts the current byte from the proper location (step 455). The building routine 335 then determines whether there is any error in the extracted byte of data (step 465) and, if it detects an error, stores error information for later processing (step 470). If it detects no error in the extracted byte of data, the building routine 335 stores the extracted byte of data as the next byte of the transmitted signal (step 475).

After processing the extracted byte of data or storing error information, the building routine 335 determines whether the last byte required for the output signal has been extracted (step 480). If the last byte has not been extracted, the building routine 335 moves to the next byte in the output signal (step 485) and returns to step 455. If the last byte of the incoming signal has been extracted, the building routine 335 determines whether any errors have been detected during the entire processing of the signal (step 490). If errors have been detected, the building routine 335 exits to an error handling routine to process the detected

errors (step 493). If no errors have been detected, the building routine 335 transmits the output signal (step 495) and ends processing.

Preferably the parsing routine 330 and the building routine 335 do not contain any information specific to the handling of any particular data format, but rely on the definition files 350 and 355 to provide all such information. Because the definition files 350 and 355 can contain information related to multiple data formats, a separate parser 310 and builder 315 or a separate parsing routine 330 and building routine 335 are not required for each format that may be used. In addition, this allows for greater flexibility for the parser 110 and builder 115 since new formats can be added to their capabilities by simply editing the definition file.

As noted above, the definition files 350 and 355 preferably contain information relating to the handling of errors, including basic information on how to process the errors prior to the execution of an error handling routine. This allows the parser 110 and builder 115 to process an entire input signal or an entire constructed output signal to detect all of the errors present in the signal before executing the error handling routine. This reduces the time necessary for processing any signal having more than one error by eliminating the need to process the signal once for every error present.

Although the present embodiment refers to input and output signals comprised of information broken up into bytes, other types of signals may also be processed. For example, the signal could

be broken up into single bits of information, multiple-byte groups, or any other desired division of the signal data.

In addition to the above descriptions, many modifications may be made to adapt a particular element, technique or implementation to the teachings of the present invention without departing from the central scope of the invention. Therefore, it is intended that this invention not be limited to the particular embodiments and methods disclosed herein, but that the invention include all embodiments falling within the scope of the appended claims.

CLAIMS

1. An automatic parser for parsing a data stream, comprising:

a memory containing a definition file;

means for receiving the data stream;

means for decomposing the received data stream into individual pieces of data based on parsing information in the definition file; and

means for transmitting the individual pieces of data.

2. The automatic parser of claim 1, further comprising a memory, wherein the transmitting means includes means for transmitting the individual pieces of data to the memory and placing the individual pieces of data into the memory based on the parsing information in the definition file.

3. The automatic parser of claim 1, further comprising a decoder for decoding the received data stream when the received data stream is encoded.

4. The automatic parser of claim 1, further comprising:

means for detecting errors in the received data stream based on error information stored in the definition file; and

means for handling the error when the error detecting means detects an error.

5. The automatic parser of claim 4, wherein the error detecting means includes means for detecting all of the errors in the received data stream before the error handling means handles any of the errors.

6. The automatic parser of claim 1, further comprising:
means for determining the format of the received data stream,
wherein the parsing information in the definition file is
further divided into a plurality of separate sets of parsing
information, each corresponding to a different possible format,
and

wherein the decomposing means decomposes the received data
stream into individual pieces of data and the transmitting means
transmits the individual pieces of data based on the determined
format and the separate set of parsing information corresponding
to the determined format.

7. An automatic builder for creating an output data stream,
comprising:

a memory containing a definition file;
means for receiving a plurality of individual pieces of data;
and

means for composing the output data stream out of the
individual pieces of data based on composing information in the
definition file.

8. The automatic builder of claim 7, further comprising a
memory containing the plurality of individual pieces of data,
wherein the receiving means receives the plurality of individual
pieces of data from the memory.

9. The automatic builder of claim 7, further comprising an
encoder for encoding the output data stream.

10. The automatic builder of claim 7, further comprising:
means for detecting errors in the plurality of individual pieces of data based on error information stored in the definition file; and

means for handling the error when the error detecting means detects an error.

11. The automatic builder of claim 10, wherein the error detecting means detects all of the errors in the plurality of individual pieces of data before the error handling means handles any of the errors.

12. The automatic builder of claim 7, further comprising:
means for determining a desired format of the output data stream,

wherein the composing information in the definition file is further divided into a plurality of separate sets of composing information, each corresponding to a different possible format, and

wherein the composing means composes the output data stream out of the individual pieces of data based on the desired format and the separate set of composing information corresponding to the desired format.

13. A method, executed by a processor, for automatically parsing an input data stream, including the steps of:

receiving the input data stream;

decomposing the received input data stream into individual pieces of data based on decomposing information stored in a definition file residing in a memory; and

transmitting the individual pieces of data.

14. The method for automatically parsing an input data stream of claim 13, wherein the transmitting step transmits the individual pieces of data into a memory based on the decomposing information.

15. The method for automatically parsing an input data stream of claim 13, further including the step of decoding the received data stream when the received data stream is encoded.

16. The method for automatically parsing an input data stream of claim 13, further including the steps of:

detecting errors in the received data stream based on error information; and

handling the error when the error detecting step detects errors.

17. The method for automatically parsing an input data stream of claim 16, wherein the error detecting step detects all of the errors in the received data stream before the error handling step handles any of the errors.

18. The method for automatically parsing an input data stream of claim 13, further including the step of:

determining the format of the received data stream,

wherein the parsing information is further divided into a plurality of separate sets of parsing information, each corresponding to a different possible format, and

wherein the decomposing step decomposes the received data stream into individual pieces of data and transmits the individual pieces of data based on the determined format and the separate set of parsing information corresponding to the determined format.

19. A method, executed by a processor, for automatically creating an output data stream, including the steps of:

receiving a plurality of individual pieces of data;

composing the output data stream out of the individual pieces of data based on composing information stored in a definition file residing in a memory; and

transmitting the composed output data stream.

20. The method for automatically creating an output data stream of claim 19, wherein the receiving step receives the plurality of individual pieces of data from a memory.

21. The method for automatically creating an output data stream of claim 19, further including the step of encoding the output data stream.

22. The method for automatically creating an output data stream of claim 19, further including the steps of:

detecting errors in the plurality of individual pieces of data based on error information; and

handling the error when the error detecting step detects an error.

23. The method for automatically creating an output data stream of claim 22, wherein the error detecting step detects all of the errors in the plurality of individual pieces of data before the error handling step handles any of the errors.

24. The method for automatically creating an output data stream of claim 19, further including the step of:

determining a desired format of the output data stream,

wherein the composing information is further divided into a plurality of separate sets of composing information, each corresponding to a different possible format, and

wherein the composing step composes the output data stream out of the individual pieces of data based on the desired format and the separate set of composing information corresponding to the desired format.

25. An automatic parser for parsing a received data stream and creating an output data stream, comprising:

a memory containing an input definition file;

a memory containing an output definition file;

means for decomposing the received data stream into a plurality of individual pieces of input data based on information in the input definition file;

means for transmitting the individual pieces of data;

means for receiving a plurality of individual pieces of output data; and

means for composing the output data stream out of the plurality of individual pieces of output data based on information in the output definition file.

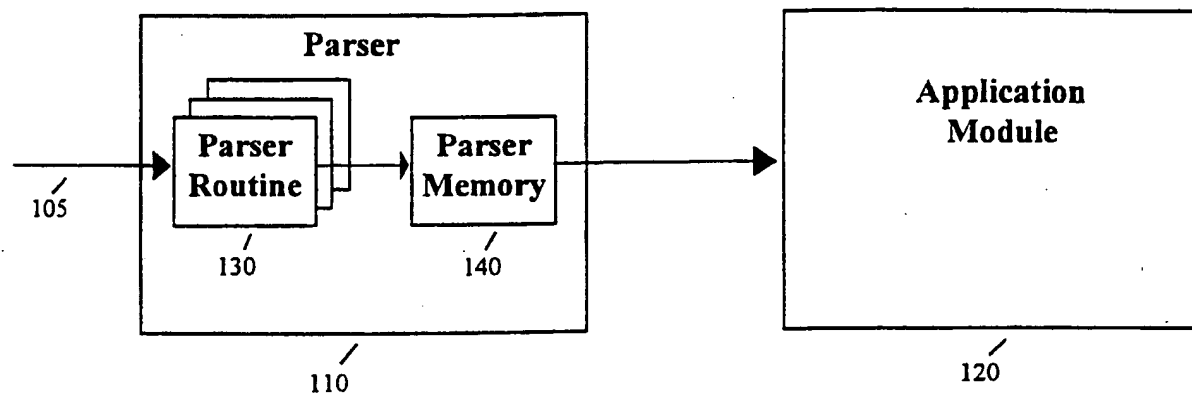
26. The automatic parser of claim 25, further comprising:

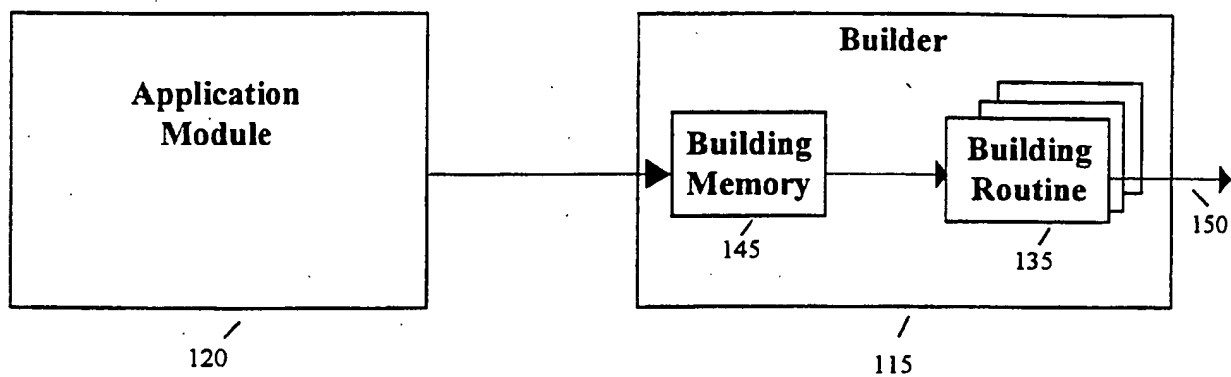
an input memory; and

an output memory containing the plurality of individual pieces of output data,

wherein the transmitting means transmits the individual pieces of data to the input memory and places the individual pieces of data into the input memory based on the parsing information in the definition file, and

wherein the receiving means receives the plurality of individual pieces of data from the output memory.

**FIG. 1A**

**FIG. 1B**

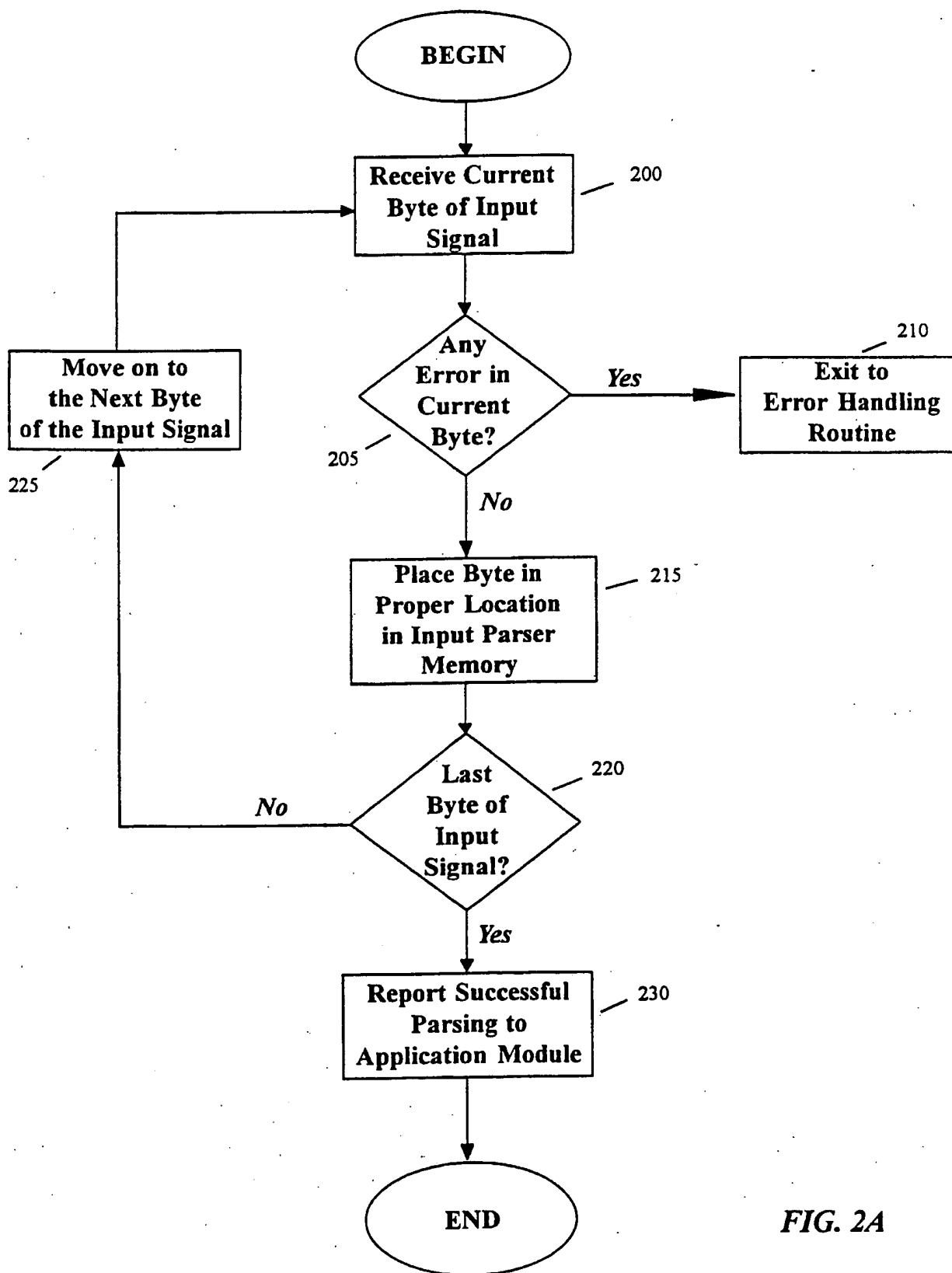


FIG. 2A

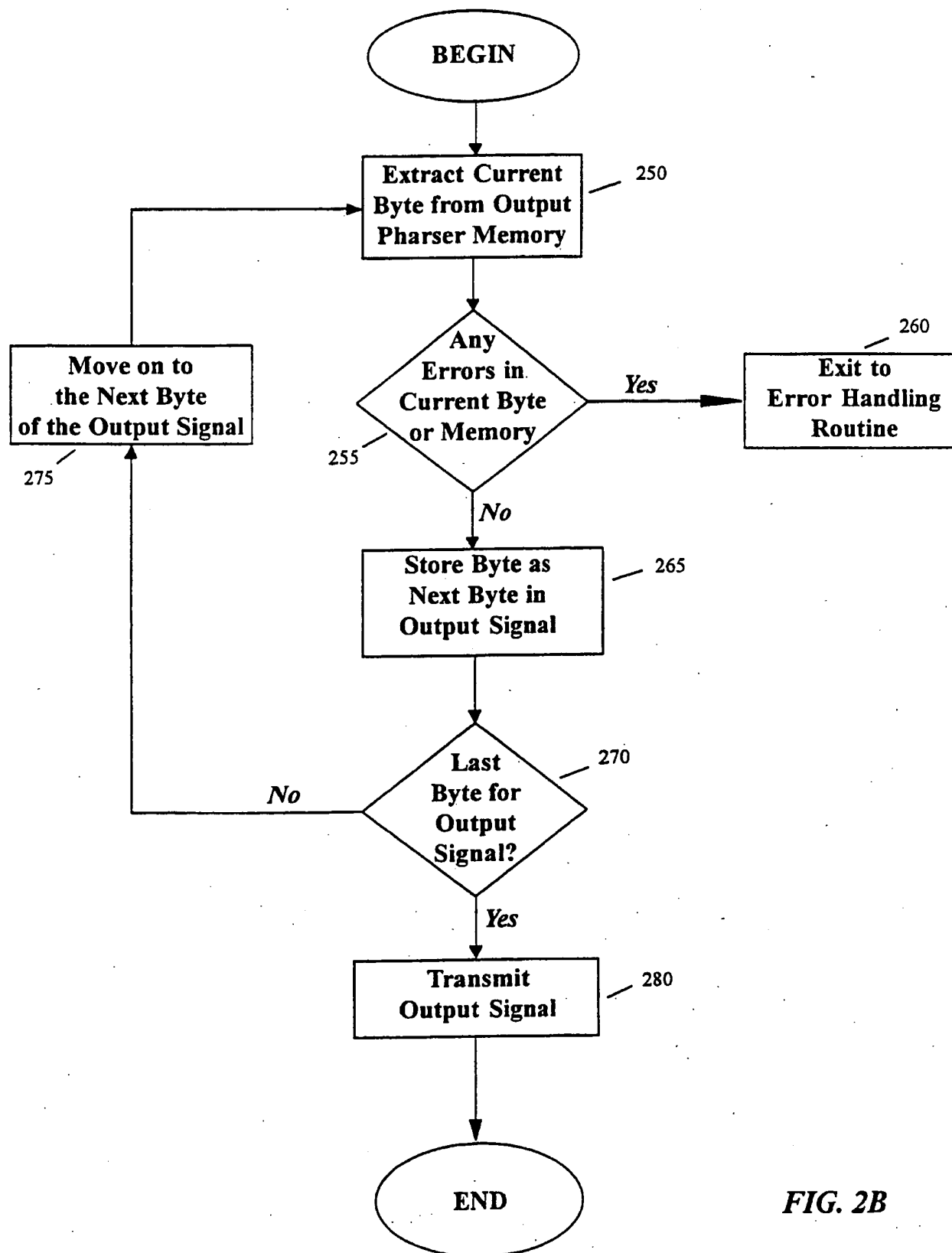
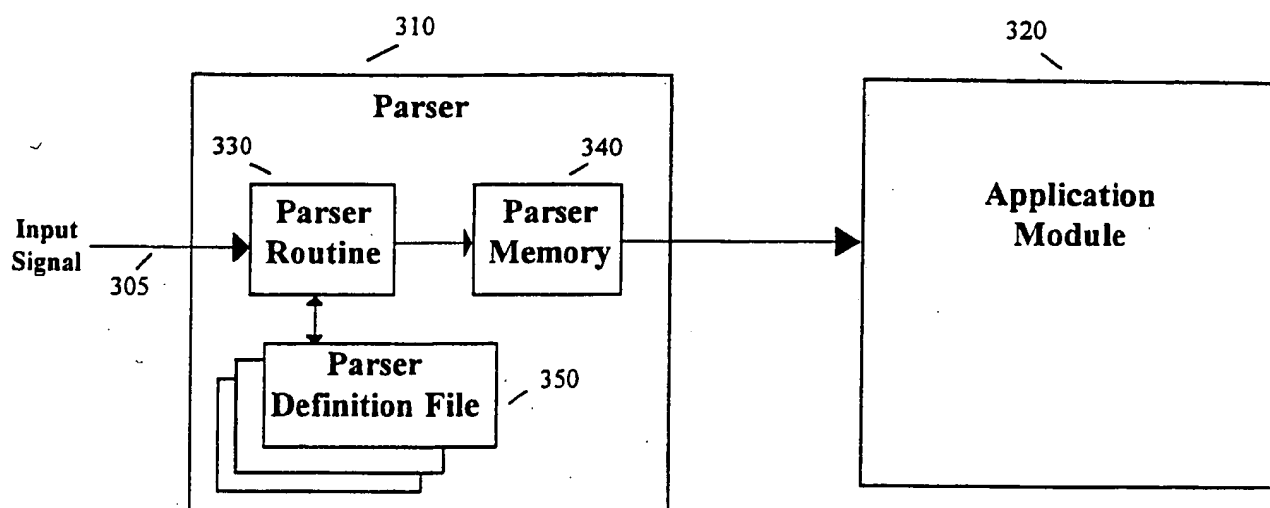
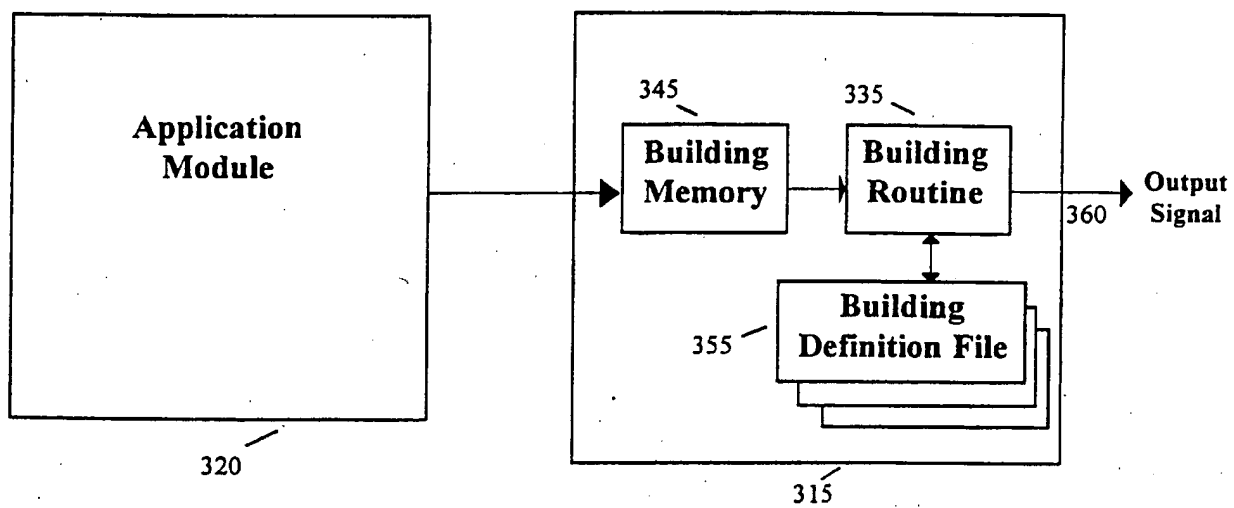


FIG. 2B

**FIG. 3A**

**FIG. 3B**

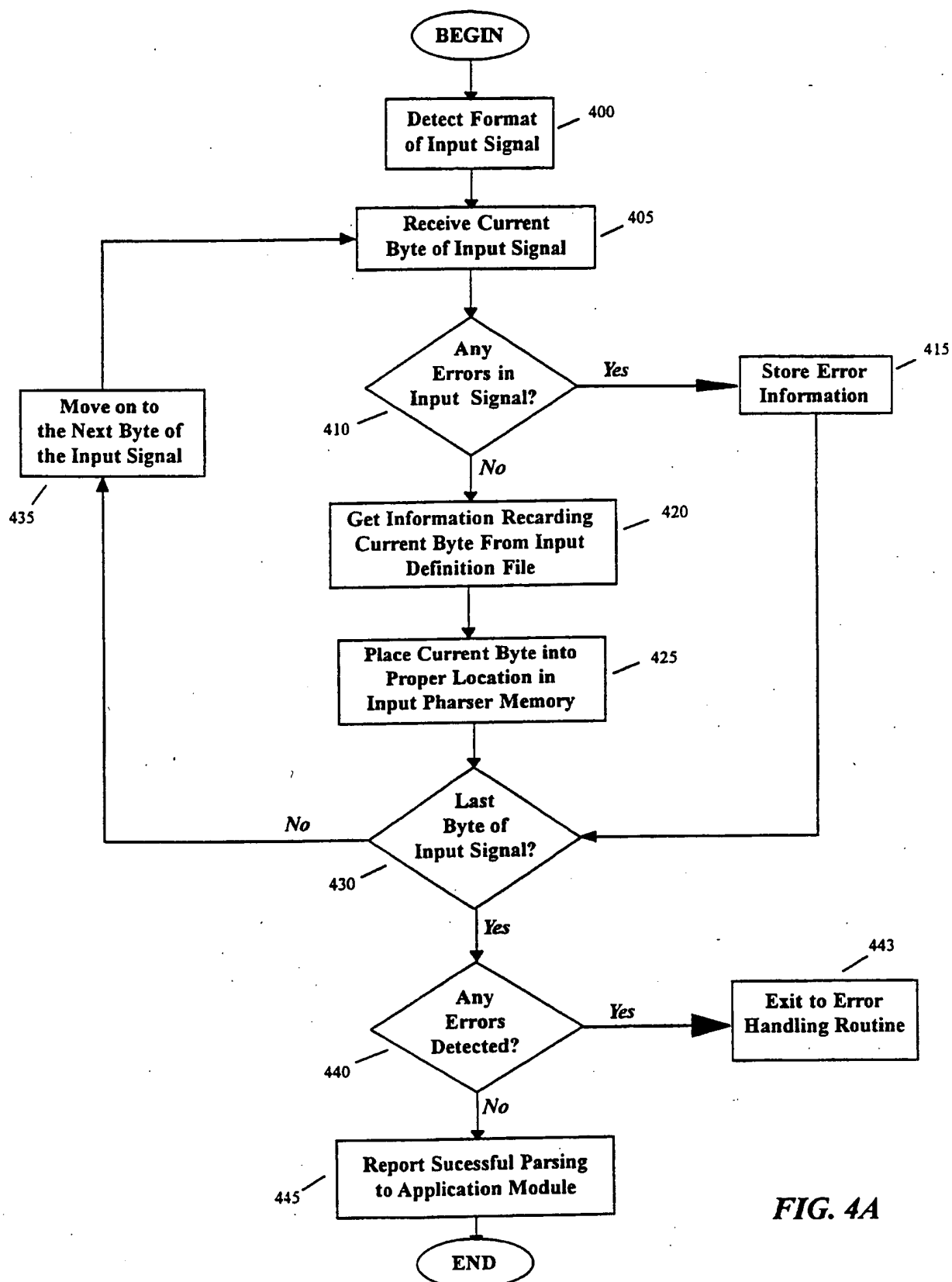


FIG. 4A

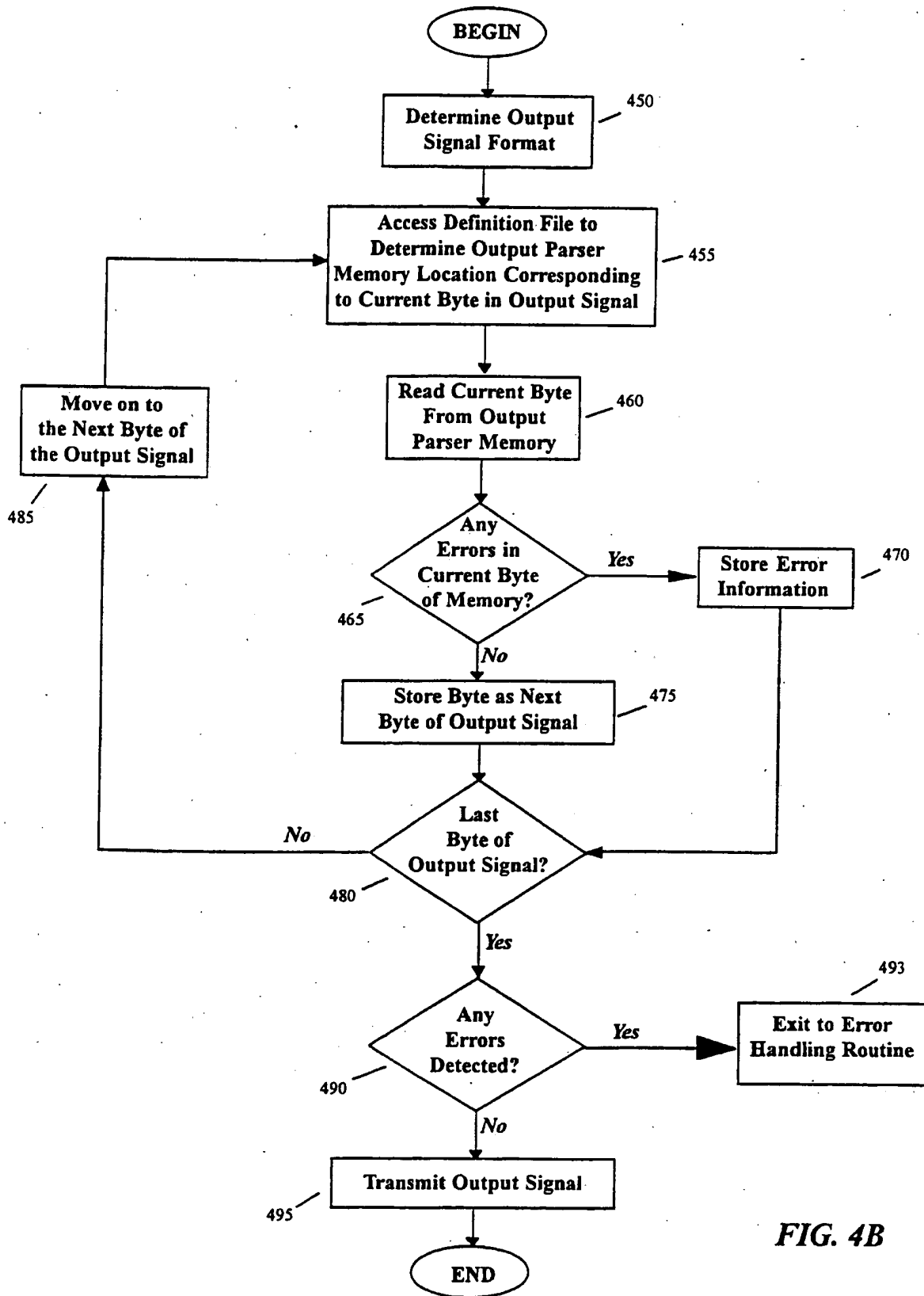


FIG. 4B

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/09821

A. CLASSIFICATION OF SUBJECT MATTER IPC(6) : G06F 9/44 US CL : 395/700 According to International Patent Classification (IPC) or to both national classification and IPC				
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) U.S. : 395/700 Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) APS search terms : encode, decode, parse, build, error correction, error detection, definition file, data format				
C. DOCUMENTS CONSIDERED TO BE RELEVANT				
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.		
Y	US, A, 5,339,421 (HOUSEL, III) 16 August 1994, See col. 1 line 16 - col. 2 line 48 and col. 5 lines 4-10, particularly col. 2 lines 10-48	1-26		
Y, P	US, A, 5,524,253 (PHAM ET AL.) 04 June 1996, See col. 10 line 33 - col. 11 line 27 particularly col. 10 lines 53-54	1-26		
Y	US, A, 5,167,034 (MACLEAN, JR. ET AL.) 24 November 1992, See col. 3 lines 21-24, lines 29-49, col. 3 line 29 - col. 4 line 4, col. 4 lines 46-56	4, 5, 10, 11, 16, 17, 22, 23		
<input type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> See patent family annex.				
<table style="width: 100%; border: none;"> <tr> <td style="width: 50%; vertical-align: top;"> * Special categories of cited documents: *A* document defining the general state of the art which is not considered to be part of particular relevance *E* earlier document published on or after the international filing date *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) *O* document referring to an oral disclosure, use, exhibition or other means *P* document published prior to the international filing date but later than the priority date claimed </td> <td style="width: 50%; vertical-align: top;"> *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art *Z* document member of the same patent family </td> </tr> </table>			* Special categories of cited documents: *A* document defining the general state of the art which is not considered to be part of particular relevance *E* earlier document published on or after the international filing date *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) *O* document referring to an oral disclosure, use, exhibition or other means *P* document published prior to the international filing date but later than the priority date claimed	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art *Z* document member of the same patent family
* Special categories of cited documents: *A* document defining the general state of the art which is not considered to be part of particular relevance *E* earlier document published on or after the international filing date *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) *O* document referring to an oral disclosure, use, exhibition or other means *P* document published prior to the international filing date but later than the priority date claimed	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art *Z* document member of the same patent family			
Date of the actual completion of the international search 27 AUGUST 1996		Date of mailing of the international search report 11 OCT 1996		
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230		Authorized officer <i>KAKALI CHAKI</i> <i>Joni Hill</i> Telephone No. (703) 305-9600		